Express Mail Label EL594561615US

DP-304678

ACTUATOR CONTROLLER FOR SELECTING A COMMUNICATION LANGUAGE

BACKGROUND

[0001] Currently, actuators are used in vehicular applications such as heavy-duty diesel trucks, particularly in turbo-charged and emission control systems. These actuators, referred to as remote smart actuators (RSA's), integrate a microprocessor-based electronic controller into a brushless motor/gear train/output shaft mechanism. The primary function of the RSA is to position an output shaft quickly and accurately as commanded by the vehicle's Engine Control Module (ECM). This action is then translated via linkage to the appropriate system actuator.

[0002] While many different RSA's are developed to perform similar functions, they often differ due to particular application requirements or due to varied manufacturer design. Therefore, although these actuators perform essentially the same function (with slight calibration differences), they often use different electronic communication methods to perform these functions. Some of these electronic communication methods or "languages" include control area network (CAN), pulse width modulation (PWM), Analog and universal asynchronous receiver/transmitter (UART) techniques.

[0003] The use of varied languages creates some complications when it comes to the testing and calibration of RSA's. The typical industry solution to this complication is to prepare a distinct software set for each different application. The nonvolatile memory of the RSA's microprocessor is loaded with a specific software set that conforms with "language" that works for that particular RSA in a particular application. Therefore, only after such special

accommodations will the RSA be able to be calibrated, function, and tested properly. This invention allows for the functionality and testing of RSA's without recalibrating for a specific communications method.

SUMMARY

[0004] An exemplary embodiment of the invention is an actuator controller for use in an actuator assembly. The actuator controller includes a processor for controlling an actuator associated with the actuator controller. A communication device is in communication with the processor. The communication device receives control commands from a master controller and is capable of communicating in multiple languages. A memory is accessible by the processor and the processor determines the language of the control commands and retrieves a control program from the memory corresponding to the language.

[0005] The above described and other features are exemplified by the following figures and detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Referring now to the Figures, which are meant to be exemplary and not limiting, and wherein like elements are numbered alike in the figures:

[0007] Figure 1 is block diagram of an exemplary actuator controller; and,

[0008] Figure 2 is a flowchart of a process for determining a command source and selecting an appropriate language.

DESCRIPTION OF THE PREFERRED EMBODIMENT

[0009] Figure 1 presents a basic representation of a controller 2 used to control and test an actuator. The controller 2 may be embedded along with the actuator in an actuator assembly. The controller 2 includes a processor 10 which may be implemented using existing microprocessors. A communication interface

20, which may include more than one type of communication input/output, provides for communication with a master controller such as an ECM. Commands for positioning the actuator are provided to controller 2 through communication interface 20. Control signals to the actuator may be sent over communications interface 20 or a separate I/O port. Memory devices such as random access memory (RAM) 30, read only memory (ROM) 40 and non-volatile random access memory (NVRAM) 50 are in communication with processor 10. It is understood that additional components (e.g., power supply, resistors, capacitors, etc.) may be associated with processor 10.

[0010] Initially, when an actuator assembly is assembled and initialized for its intended application, various parameters relating to that particular actuator assembly will be loaded into nonvolatile memory (NVRAM). For example, the actuator may be driven through its full range of operation in order to establish its "full close" and "full open" positions. Such positions define the full range of motion of the actuator. Positions within that range are then scaled into a percentage value based on those two fundamental positions. These parameters are stored as bytes in NVRAM for later use during operation and testing. Such information is used to calibrate the particular actuator being used, so that the system will execute commands that accurately translate into the intended behavior by the actuator. One calibration parameter relates to the communication format, referred to herein as the "language," being used to provide input/output to processor 10. The differences in languages can significantly affect the accurate translation of signals to and from the processor.

[0011] Additionally, as calibration parameters of many recognized or commonly used actuators are well known, such parameters can be pre-programmed into NVRAM 50. Thus, an actuator identifier, such as a part number or manufacturer code, may be used to designate which calibration parameters are loaded into NVRAM 50 at the time of initialization. The calibration parameters may include the communication language used for that actuator. This initialization technique can save time by providing information about the intended primary

language for the chosen actuator. By knowing the intended primary or secondary language used for an actuator, signaling commands can be adjusted or calibrated so that it functions properly.

[0012] An exemplary embodiment of the invention provides a single software set that contains parameters for a variety of actuators and also provides a method so that the software initialized into NVRAM can identify which parameters to use. Figure 2 depicts a process implemented by processor 10 in response to a computer program stored, for example, in NVRAM 50. To verify that the controller continually uses the proper parameters, the process as demonstrated in Figure 2 may be repeated on a regular basis. For example, a preferred embodiment reiterates this procedure every 3 milliseconds.

[0013] Referring to Figure 2, the process begins at step 60 where processor 10 checks for a calibrated source, and also checks whether that source is actively present in the system and generating proper values. Only a "yes" response to all of these questions will generate a "yes" for step 60. As discussed above, a calibrated source would be one that has been identified and registered in NVRAM 50 along with its parameters, or is predefined in NVRAM and identified by manufacturer code. If a calibrated source is identified, then the system will check whether that source is active. In other words, is processor 10 receiving signals from that control command source. Next, processor 10 confirms that the control commands are valid or consistent with expected values for that actuator. For example, if a control command was received that attempts to position an actuator beyond its intended range, such a value would not be valid. Only a positive response to all three questions in step 60 will generate a yes response, otherwise a no response is generated.

[0014] A positive response at 60 means that there has been a calibrated, active and valid control command detected. In such a case, steps 100, 200, 300 and 400 determine the language of the source control commands. Where an actuator has been calibrated, the language is typically known and may be identified by a language identifier stored in NVRAM 50. Processor 10 accesses the

language data from NVRAM 50 to determine the appropriate control command language. Once the language is detected, processor 10 designates that mode of communication via communications interface 20 for accurate operations. Therefore, if processor 10 detects CAN language at step 100, the processor selects CAN communications at step 175. Likewise, detecting UART at step 200 leads to selection of UART communications at step 275, detecting PWM at step 300 leads to selecting PWM communications at step 375, and detecting analog at step 400 leads to selecting analog control at step 475. The application continues to step 550 and the cycle will run periodically (e.g., every 3 milliseconds).

[0015] A negative response to any of the questions presented in step 60 means that there either is no language identifier stored in NVRAM 50 or that there is a system malfunction. In such a scenario, the source language is detected by processor 10 in order to properly translate control commands. Thus, step 150 initiates a hierarchical test in order to determine which language to use. In particular, at step 150 the processor checks to determine if the CAN language being received and whether the signals being received are valid. Such a check can be done in various ways, such as verifying communication characteristics (e.g., headers, interrupts, framing errors, baud rate, and/or the addressing identification on the printed wiring board). Once again, the "active" aspect of the check relates to whether a CAN signal is being received, while the "valid" aspect relates to whether the signal is appropriate for the application. A positive response to both these questions will designate CAN as the command source language at step 175. Processor 10 then retrieves the corresponding CAN control program from NVRAM 50 so that the CAN control commands are properly processed by processor 10. A negative response to either of the questions at step 150 continues the hierarchical test for the next priority language.

[0016] At step 250, processor 10 determines if a UART signal is active and valid. A UART signal can be identified based on communication characteristics in much the same way as a CAN signal. The primary distinction between these two languages is the scaling. Due to the nature of a UART interface,

an all "0" or all "1" byte value is not desirable. Therefore, an offset is provided in the scaling to avoid such values and have them result in a non-valid signal. Also, as with CAN, an active and valid UART signal will result in the selection of UART at step 275 as the command source language.

[0017] The next language checked is PWM at step 350 and then analog at step 450 if PWM is not present. Both languages may be detected based on known communication characteristics. Once again an active and valid signal from these sources will designate the corresponding language at steps 375 and 475, respectively. It should be noted that the actuator will be provided with what is referred to as a "duty cycle," when using PWM or analog communications due to generally accepted tolerances. A duty cycle will operate the actuator in a range, such as 5% to 95%. As with other predefined calibration parameters, the duty cycle parameters would be stored in NVRAM 50.

[0018] While Figure 2 designates a particular priority for the languages that are considered, it is understood that a different order can be defined in the initialization software. What is more, the software may be provided with the capability of receiving configuration input in order to change the language priority used in the procedure described by Figure 2. Certain changes to bit designations in NVRAM may perform this function. Also, some circuits do not allow analog communication. In order to deal with such cases, initial parameters may be set to make step 450 automatically fail, thus not allowing the selection of analog communications. Conversely, the initial parameters may be designed to not accept analog selection unless it has been specifically enabled in NVRAM.

[0019] Typically, only one source will be provided on any actuator in application. There may be applications where a secondary source will become available only if the primary source has failed.

[0020] Finally, in the event that all the tests 150, 250, 350, 450 discussed above fail, the system will direct the actuator into a command default position 500. Such a position would be stored in NVRAM as part of the initial

calibration, and would be a signal to a tester or user that something is wrong with the system.

[0021] The ability of the controller to communicate in multiple formats simplifies the job of the application/system engineers by allowing them to read/write control information to/from any actuator assembly via UART regardless of the default communication programmed in NVRAM. The controller detects the UART communications as shown in step 200 and selects UART communications mode at step 275 as described above with reference to Figure 2.

[0022] While preferred embodiments have been shown and described, various modifications and substitutions may be made thereto without departing from the spirit and scope of the invention.

What is claimed is: